Computer Science Competition
Invitational A 2024
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Akansha
Problem 2	Andrew
Problem 3	Bautista
Problem 4	Chuanli
Problem 5	Dominika
Problem 6	Gerald
Problem 7	Hiro
Problem 8	Joan
Problem 9	Manisha
Problem 10	Patryk
Problem 11	Rodrigo
Problem 12	Svetlana

1. Akansha

Program Name: Akansha.java   **Input File: None**

This is the year!!! The time is now!!! Leaguetown HS is ready to go to state in UIL Computer Science. Akansha is the coach and she is determined to work as hard as she can to get her students ready for the contest season.

In order to do this, she is making a poster with the 2023 UIL Computer Science State Champions. She is printing the conference, the team score, and the school from all six State Champions.

Will you help her? Write a program to print the chart below exactly as you see in the sample. With your help. Leaguetown HS may be on the verge of making history.

Input: None

Output: Your chart will have seven lines - a title line followed by six lines including the conference, the score, and the school name. The output should be formatted exactly as shown below. Note that all letters are uppercase.

Sample input:
```
None
```

Sample output:
```
2023 UIL COMPUTER SCIENCE STATE CHAMPIONS
1A  501 ASPERMONT
2A  838 SAN AUGUSTINE
3A  921 FORT WORTH HARMONY INNOVATION
4A 1026 NEEDVILLE
5A 1193 FRISCO LEBANON TRAIL
6A 1186 ALLEN
```

2. Andrew

Program Name: Andrew.java   **Input File: andrew.dat**

Andrew is simply mesmerized by one-digit natural numbers. He has spent countless hours … hmm ... counting from one to nine. Numbers ten and greater do not impress him. He does not even think about numbers less than one (as he is a very positive person). Don't even attempt to give him a decimal number.

Andrew wants the user to input a natural number. He would like a program to list the one-digit factors of the number written in word form. Please write the program that will do just that.

Input: The first line of data will be an integer in the range [1,20] indicating the number of data sets. Each data set will contain one integer in the range [1,1000000].

Output: Each output line will contain one or more of the words {ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT, NINE} separated by one space. The words represent the one-digit factors. These should be written in numerical order where ONE is first and NINE is last.

Sample input:
```
5
7
100
63
42
2520
```

Sample output:
```
ONE SEVEN
ONE TWO FOUR FIVE
ONE THREE SEVEN NINE
ONE TWO THREE SIX SEVEN
ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE
```

3. Bautista

Program Name: Bautista.java   **Input File: bautista.dat**

Bautista had a traumatic learning experience when he was a child. The week that his elementary school class was learning about vowels was not a good week for him. He had a terrible toothache all week and it caused him extraordinary pain. Now, whenever he encounters a vowel, he feels the same pain he felt so long ago in that classroom. He thinks, "OUCH!!!"

Your job is to write a program to help him visualize his pain. Allow the user to input a word. Replace each vowel with "**OUCH!!!**" As you do this, make sure there is no white space separating any part of each output.

Input: The first line of data will be an integer in the range [1,20] indicating the number of data sets. Each data set will contain one word written in lowercase letters with no spaces and no punctuation. Words will have a length in the range [1,20].

Output: Each output line will contain lowercase consonants and OUCH!!!'s. "y" will <u>not</u> be considered a vowel. There should be no white spaces between characters in your output.

Sample input:
```
6
university
aardvark
brrr
computer
science
aaaaa
```

Sample output:
```
OUCH!!!nOUCH!!!vOUCH!!!rsOUCH!!!ty
OUCH!!!OUCH!!!rdvOUCH!!!rk
brrr
cOUCH!!!mpOUCH!!!tOUCH!!!r
scOUCH!!!OUCH!!!ncOUCH!!!
OUCH!!!OUCH!!!OUCH!!!OUCH!!!OUCH!!!
```

4. Chuanli

Program Name: Chuanli.java   **Input File: chuanli.dat**

Your friend Chuanli accidentally drank backwards potion, so now he writes everything backwards! This is really bad timing, as he has a paper due tomorrow. You need to write a program to reverse the order of all the words in his paper, as well as reversing all the letters in each word, so that Chuanli will pass.

Input:
The input will begin with an integer, n (0 < n <= 1000), denoting the number of test cases to follow. Each test case will consist of an unknown number of space-separated strings to be reversed. These strings can be made up of any ASCII character, and each test case will appear on its own line.

Output:
Output the lists of strings, with the order of the strings reversed, and the order of the characters in each of the strings reversed as well, with all the words being separated by spaces. Each test case output should appear on its own line.

Sample input:
```
3
eM stI olleH
snoitaluclac ym ot gnidroccA
A na em eviG syaS ilnauhC
```

Sample output:
```
Hello Its Me
According to my calculations
Chuanli Says Give me an A
```

5. Dominika

Program Name: Dominika.java   **Input File: dominika.dat**

An equilateral triangle is a triangle in which all three sides are the same length. An equilateral triangle is also equiangular, meaning that all three interior angles are congruent and are each 60 degrees. For example, if the points (7,-6) and (1,2) are given, two equilateral triangles can be formed. The first triangle (dashed lines) is formed with the addition of the point A (-2.93,-7.2), and the second triangle (dotted lines) is formed with the addition of the point B (10.93,3.2). Note: these values are rounded to two decimal places.
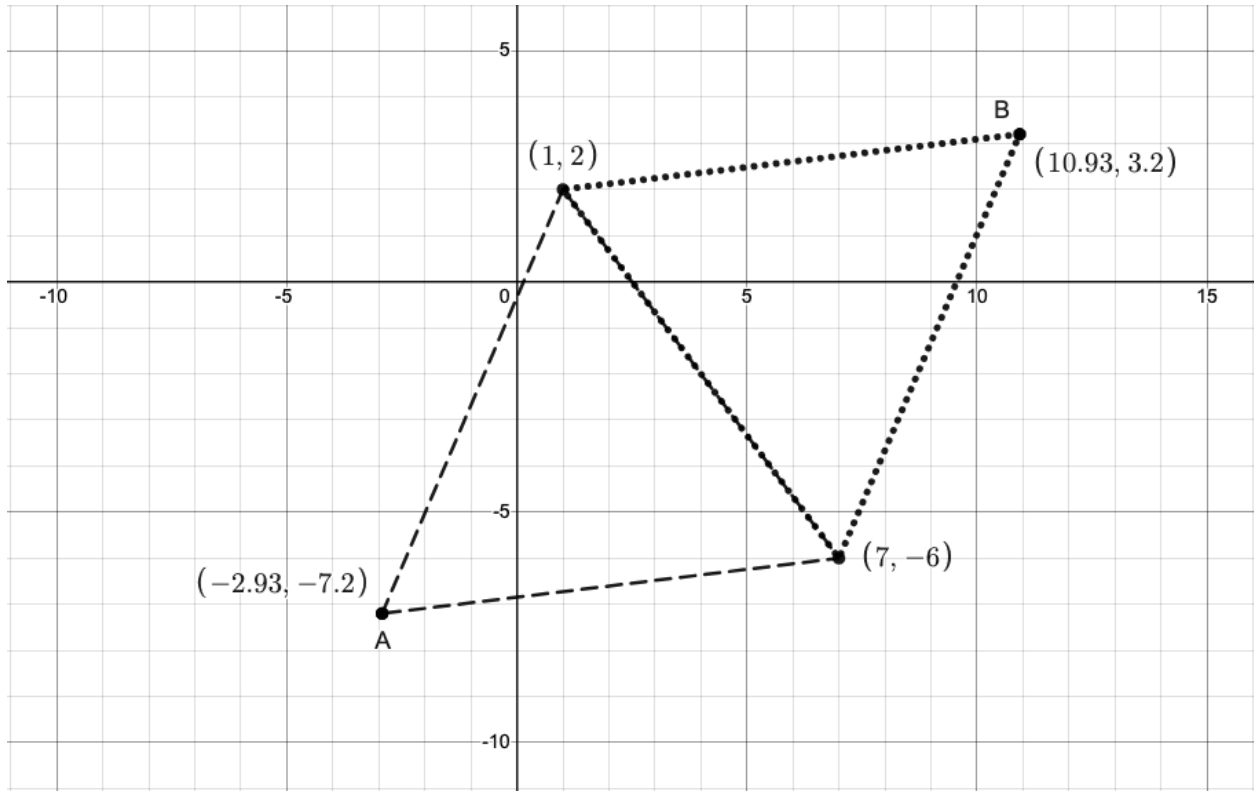
Dominika has been given two coordinates representing two out of the three points on an equilateral triangle, but the third point is missing. Can you help Dominika write a program that is able to find these two missing points?

Input: First line will contain an integer N in range [1,20], the number of test cases. Each of the following N lines will contain four whole numbers, separated by spaces. These numbers represent Ax Ay Bx By, where A and B are the two given points. All whole numbers will be in range [-20,20].

Output: For each test case you are to output: "Test case: #" where # is the current test case, followed by the two points that are missing, that are needed to make the two equilateral triangles. Each point is to be printed out on its own, individual line. All output should be rounded and printed out to two decimal places. The point with the smallest x value should be output first. If both x coordinates are the same, then output the point with the smallest y value.

Sample Input and Output on next page

Dominika continued…

Sample input:
```
10
4 -7 7 14
-10 -14 -1 2
9 8 -18 5
7 -6 1 2
-15 -1 -14 15
1 -16 -13 11
-5 4 8 -13
-6 -2 8 18
3 -16 1 -12
0 0 3 0
```

Sample output:
```
Test case: 1
(-12.69,6.10)
(23.69,0.90)
Test case: 2
(-19.36,1.79)
(8.36,-13.79)
Test case: 3
(-7.10,29.88)
(-1.90,-16.88)
Test case: 4
(-2.93,-7.20)
(10.93,3.20)
Test case: 5
(-28.36,7.87)
(-0.64,6.13)
Test case: 6
(-29.38,-14.62)
(17.38,9.62)
Test case: 7
(-13.22,-15.76)
(16.22,6.76)
Test case: 8
(-16.32,20.12)
(18.32,-4.12)
Test case: 9
(-1.46,-15.73)
(5.46,-12.27)
Test case: 10
(1.50,-2.60)
(1.50,2.60)
```

6. Gerald

Program Name: Gerald.java   **Input File: gerald.dat**

Gerald is just beginning to learn programming and still stumbles at times. He has a collection of data to use while practicing his programming skills. For now, he would like to extract some basic characteristics of that data like the smallest and largest numbers and the mean or average.

Can you help Gerald?

Input: An unknown number of lines containing whitespace-separated integers in the range [1000, 9999]. There will be no more than 100 integers.

Output: Just four items, the count of integers that were input, the smallest and largest integers found in the data and the mean or average displayed with exactly one digit after a decimal point. Label and format the results exactly as shown below in the sample output.

Sample input:
```
5867 4237 9708 9677 4971 2699 6391 7321 7171
7073 2882 5315 2708 7861 1856 9658 5354
6305 1764 9054 5554 7846 6725 8234 2305 7672 6042
1827 1882 9111 2406 5853 5911
7843 1410 3959 6975 2491 2331 9795 2855 7434
1262 1120 8278 2227 9911 3328 5967 9607 9878 1326
7138 4192 5937 2893 9403 2452 6843
2821 6871 5424 5376 3834 5234 8013 3037
1175 3983 2095 7873 8505
4781
```

Sample output:
```
COUNT:73
SMALLEST:1120
LARGEST:9911
MEAN:5412.6
```

7. Hiro

Program Name: Hiro.java   **Input File: hiro.dat**

Surprise! For your birthday your mom got you a roman soldier named Hiro who can help with your math homework. The only issue is, all his answers are in roman numerals. You need to write a program to convert all his roman numeral answers into decimal numbers so you can complete the math homework. Roman numerals are listed as uppercase letters, with each letter corresponding to the following values:

```
I - 1
V - 5
X - 10
L - 50
C - 100
D - 500
M - 1000
```

All roman numerals will be listed in descending order by value, except for the special case when a lower value character is listed before the character with a greater value (such as IV, which is the value 4), we subtract the lower value from the higher value.

Input:
The input will begin with an integer, n, denoting the number of test cases to follow. Each test case will consist of a string of roman numerals to be converted. Each string will contain only uppercase alphabetic ASCII characters, each case will appear on its own line.

Output:
Output the integer value that is equivalent to the string of roman numerals given.

Sample input:
```
3
MCMIV
XIVII
IVI
```

Sample output:
```
1904
16
5
```

8. Joan

Program Name: Joan.java   **Input File: joan.dat**

Joan really enjoys her programming class but her true passion is the English language. However, programming has caused her to view English writings somewhat differently and she realizes that words are simply pieces of data put together carefully to create a desired meaning. For now, she has decided to practice her programming skills by creating a program to perform a very simple analysis of relatively small pieces of prose.

Joan has big plans for analyzing written pieces but decided to start with a program that simply counts the number of words in a sample of prose and calculate the average length of all words.

Input: A sample of English prose with an unknown number of lines containing an unknown number of whitespace-separated words on each line. The number of lines will be in the range [2,25]. Words may consist of both uppercase and lowercase letters and there are no non-letter symbols or numbers.

Output: A single line containing the number of words and the calculated average length of all words, formatted exactly as shown below in the sample output. The average length is rounded to the nearest whole number.

Sample input:
```
Joan really enjoys her programming class but her true passion is the English language
However programming has caused her to view English writings somewhat differently
and she realizes that words are simply pieces of data put together carefully to
create a desired meaning For now she has decided to practice her programming skills
by creating a program to perform a very simple analysis of relatively small pieces of
prose
Joan has big plans for analyzing written pieces but decided to start with a program
that simply counts the number of words in a sample of prose and calculate the
average length of all words
Input A sample of English prose with an unknown number of lines containing an
unknown number of whitespace separated words on each line The number of lines will be
in the range Words may consist of both uppercase and lowercase letters and there are
no nonletter symbols or numbers
Output A single line containing the number of words and the calculated average length
of all words formatted exactly as shown below in the sample output The average length
is rounded to the nearest whole number
```

Sample output:
```
189 words with an average length of 5 letters
```

9. Manisha

Program Name: Manisha.java   **Input File: manisha.dat**

Manisha is preparing for her boolean algebra exam where she will need to show mastery in her ability to write boolean equations from their equivalent digital electronics symbols. More specifically, Manisha will need to write several sums of products. A valid sum consists of one or more variables (denoted by a single uppercase letter) with a possible negation tied to an individual variable (denoted by the character '/').

While Manisha is confident in her abilities to derive those boolean equations, she is not so confident in her ability to format her answer properly for the auto-grader to mark her answer as correct. For an answer to be marked as correct, it needs to be formatted as follows:

1. Within a given product, products should be sorted…
    a. First by increasing alphabetical order.
    b. Second by the presence of a negated relation.
        1) I.e., `A` comes before `/A`
2. Between different products, products should be sorted…
    a. First by increasing length.
    b. Second by increasing alphabetical order.
    c. Third by the presence of a negated relation.
        1) I.e., `ABC` comes before `A/BC`
3. Extraneous spaces should be removed from the equation as a whole…
    a. Within a given product, there should not be any spaces between symbols.
    b. Between any two given products there should only be a single + with no spaces on either side.

Help Manisha by writing her a program that will format her answer in the format mentioned above.

Input:
The first line of input will consist of a single integer n ($1 \leq n \leq 100$) denoting the number of testcases to follow. The next n lines will each contain a single string indicating the sum of products that Manisha wishes to format properly. It is guaranteed that each product is fully simplified and therefore a given variable name will **not** appear more than once within a given product.

Output:
For each of Manisha's n requests, on their own line, print a single string which is formatted as described above.

Sample input:
```
3
FEDA
ABCD+BCDE+/CDEF+CDEF+/ABCD+A/BCD
ABCD   + BCDE +    /CD EF   + CDEF+/ABCD+   A/BCD+A+B+CD+PA
```

Sample output:
```
ADEF
ABCD+A/BCD+/ABCD+BCDE+CDEF+/CDEF
A+B+AP+CD+ABCD+A/BCD+/ABCD+BCDE+CDEF+/CDEF
```

10. Patryk

Program Name: Patryk.java   **Input File: patryk.dat**

Patryk needs help with an unusual sorting problem. He has a list of 5-digit numbers and needs them sorted so the numbers are sorted from right-to-left instead of a normal left-to-right order.

Given numbers: `53334 82529 47619 68398 90010 23633 52118 49703 62205 56691 73206`

Ascending sort: `23633 47619 49703 52118 53334 56691 62205 68398 73206 82529 90010`
Descending sort: `90010 82529 73206 68398 62205 56691 53334 52118 49703 47619 23633`

"Reverse-digit"
Ascending sort: `90010 56691 49703 23633 53334 62205 73206 62258 48458 47619 82529`

The "Reverse-digit" sorted list starts with all numbers that end with `0` and following numbers are increasing in that position finishing with those that end with `9`. The third and fourth numbers both end with `3`, so they are sorted by the next digit to left and, when that digit is the same as in `48458` and `62258`, the next digit establishes the correct order. The same process is used when the remaining digits match, the digit to left determines their correct order.

Can you help Patryk solve this sorting problem?

Input: First line will contain an integer in range [5,50] which is the number of integers that follow remaining lines of data, separated by whitespace. All those integers are 5-digit numbers in range [10000, 99999].

Output: A vertical list of the "reverse-digit" sorted numbers as described above.

Sample input:
```
11
53334 82529 47619 48458 90010 23633
62258 49703 62205 56691 73206
```

Sample output:
```
90010
56691
49703
23633
53334
62205
73206
62258
48458
47619
82529
```

11. Rodrigo

Program Name: Rodrigo.java   **Input File: rodrigo.dat**

You and your best friend Rodrigo have decided to travel together. Rodrigo comes from money so you guys have some funds to travel with, however you need to determine where you can go with the amount of money you have, and which of those places you'd be most excited to go to. Rodrigo's parents would like you guys to learn something from this trip, so they have added some interesting stipulations if you are going to go:

- You have unlimited money when it comes to spending money and hotels.
- You will be given a money limit for travelling.
- You are only allowed to take planes from place to place.
- You will always begin in Dallas, but you can end anywhere (don't worry about the return journey).
- You need to show them a sorted list of the places you can afford to go to, in the order in which you are excited to go. The list will be sorted by the following criteria:
  1. First sort the list based on the number of attractions, in descending order, then
  2. Sort by the culinary rating of the city, in descending order, then
  3. Sort by the price of staying, in ascending order, then
  4. Sort by the tourist rating of the city, in descending order, then
  5. Sort the cities alphabetically.

Input: The input will begin with two integers, `c` and `f`, denoting the number of cities and flights, respectively, followed by an integer `m` denoting the amount of money you and Rodrigo have for flights. Each of the following `c` lines will contain a city listing, consisting of the following values: a string `name` denoting the name of the city, a decimal value `trtg` denoting the tourist rating of the city, a decimal value `price` denoting the price to stay in the city, an integer value `att` denoting the number of attractions, and a decimal value `crtg` denoting the culinary rating of the city, all separated by spaces. After this, the next `f` lines will each contain a flight listing, consisting of 2 strings, `a` and `b`, denoting the cities where the flight will go to and from (can go in either direction), followed by `price`, a decimal value denoting the price of the flight. The city of Dallas will always be included in the flights, but not in the listed cities.

Output: Output a numbered ranked list of the city names that you are most excited to visit, in the format shown in the Sample output section below, with the number of the city in the list first, followed by a colon, followed by a space, followed by the name of the city. To clarify: the cities in the list are any cities you can possibly visit on ANY path, not necessarily cities all on the same path. If you can get to Milan and Prague, or Paris, on 2 separate paths, all 3 should be included.

Sample input:
```
5 8 2000
Prague 4.3 321.43 9 2.7
Milan 3.7 456.70 9 4.2
Casablanca 2.6 134.52 5 3.8
Paris 1.2 1000.03 10 2.5
Beijing 3.5 300.54 7 3.9
Prague Milan 400
Milan Casablanca 700
Prague Paris 550
Casablanca Paris 700
Dallas Paris 1200
Dallas Casablanca 1100
Milan Beijing 1800
Casablanca Beijing 2000
```

Sample output:
```
1: Paris
2: Milan
3: Prague
4: Casablanca
```

12. Svetlana

Program Name: Svetlana.java   **Input File: svetlana.dat**

Svetlana's favorite activity growing up was the classic board game, Checkers. Checkers is a turn-based game played on an 8 x 8 grid between two players, Team Red and Team Black. Each player begins with 12 pieces, staggered on the dark portions of the board. Non-king pieces, typically called "men", denoted by a lowercase letter, can move diagonally forward only, one square at a time, onto an empty adjacent dark square. They capture opponent's pieces by jumping over them forward diagonally if there is an empty square immediately beyond the jumped piece. As soon as a non-king piece makes it the full distance of the board, the piece is transformed into a King, denoted as an uppercase letter, allowing it to move BOTH forward and backward! (Remember, non-king pieces can only move diagonally forward.) Red's home position will always be top of board with black's home position at bottom.

The goal of checkers is to capture all of the opponent's pieces, without losing all of your own pieces. As soon as one player is out of pieces, the game is over.

Svetlana needs your help determining how many unique **<u>single</u>** moves each team, Team Red and Team Black, can make in a given board. Although some rules allow for a double jump, Svetlana is only considering **single** jumps.

For example, given the below board state:

	0	1	2	3	4	5	6	7
0		r		r		B		r
1	r		r				r	
2		r						r
3			R					
4				b				
5			b					
6		b		b				
7	b		b		b		R	

The legal moves are as follows:

Red valid single moves: 10
(0,3) → (1,4)
(1,2) → (2,3)
(1,6) → (2,5)
(2,1) → (3,0)
(2,7) → (3,6)
(3,2) → (4,1)
(3,2) → (5,4)
(3,2) → (2,3)
(7,6) → (6,5)
(7,6) → (6,7)

Black valid single moves: 6
(0,5) → (1,4)
(4,3) → (3,4)
(5,2) → (4,1)
(6,1) → (5,0)
(6,3) → (5,4)
(7,4) → (6,5)

Continued on next page…

Svetlana continued…

Input: Input will begin with an integer N, the number of test cases. N is in the range [1,10]. Each of the following N test cases will start with 16 dashes to serve as a divider between each of the boards. Following the dashes will be 8 rows representing the checkers board. Each valid space will be either a 'r', 'R', 'b', 'B', or '_'. Where 'r' denotes a red non-king, 'R' denotes a red King, 'b' denotes a black non-king, and 'B' denotes a black King. The underscore denotes the square is empty. All, individual squares will be separated by a space to simply the reading of the input.

Output: For each of the test case, you are to output "Test case: #" where # is the current test case. On the subsequent line you are to output "Red valid single moves: RR" where RR denotes the number of valid single moves available for red to make. On the next line, you are to output "Black valid single moves: BB" where BB denotes the number of valid single moves available for black to make. To reiterate, no double jumps are allowed.

Sample input:
```
4
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ _ _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ b _ _ _ _ _
_ b _ b _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ _ _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ b _ _ _ b _
_ b _ b _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ R _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ b _ _ _ _ _
_ b _ b _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ _ _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ b _ _
_ _ b _ _ _ _ _
_ _ _ b _ _ _ _
b _ b _ b _ R _
```

Sample output:
```
Test case: 1
Red valid single moves: 10
Black valid single moves: 6
Test case: 2
Red valid single moves: 10
Black valid single moves: 8
Test case: 3
Red valid single moves: 14
Black valid single moves: 6
Test case: 4
Red valid single moves: 10
Black valid single moves: 9
```

UIL Computer Science Competition

Invitational A 2024

<u>JUDGES PACKET - CONFIDENTIAL</u>

I. Instructions

1. The attached printouts of the judge test data are provided for the reference of the contest director and programming judges. Additional copies may be made if needed for this purpose.

2. This packet must remain CONFIDENTIAL. Additional copies may be made and returned to schools when other confidential contest material is returned.

II. Table of Contents

Problem #1
60 Points

1. Akansha

Program Name: Akansha.java   **Input File: None**

Test Input File:
```
None
```

Test Output To Screen:
```
2023 UIL COMPUTER SCIENCE STATE CHAMPIONS
1A  501 ASPERMONT
2A  838 SAN AUGUSTINE
3A  921 FORT WORTH HARMONY INNOVATION
4A 1026 NEEDVILLE
5A 1193 FRISCO LEBANON TRAIL
6A 1186 ALLEN
```

Problem #2
60 Points

2. Andrew

Program Name: Andrew.java   **Input File: andrew.dat**

Test Input File:
```
10
7
100
63
42
2520
1
23
72
60
625
```

Test Output To Screen:
```
ONE SEVEN
ONE TWO FOUR FIVE
ONE THREE SEVEN NINE
ONE TWO THREE SIX SEVEN
ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE
ONE
ONE
ONE TWO THREE FOUR SIX EIGHT NINE
ONE TWO THREE FOUR FIVE SIX
ONE FIVE
```

Problem #3
60 Points

3. Bautista

Program Name: Bautista.java   **Input File: bautista.dat**

Test Input File:
```
10
university
aardvark
brrr
computer
science
aaaaa
ouch
uil
champion
contest
```

Test Output To Screen:
```
OUCH!!!nOUCH!!!vOUCH!!!rsOUCH!!!ty
OUCH!!!OUCH!!!rdvOUCH!!!rk
brrr
cOUCH!!!mpOUCH!!!tOUCH!!!r
scOUCH!!!OUCH!!!ncOUCH!!!
OUCH!!!OUCH!!!OUCH!!!OUCH!!!OUCH!!!
OUCH!!!OUCH!!!ch
OUCH!!!OUCH!!!l
chOUCH!!!mpOUCH!!!OUCH!!!n
cOUCH!!!ntOUCH!!!st
```

Problem #4
60 Points

4. Chuanli

Program Name: Chuanli.java   **Input File: chuanli.dat**

Test Input File: (*indented lines are continuation of previous line***)**
```
11
eM stI olleH
snoitaluclac ym ot gnidroccA
A na em eviG syaS ilnauhC
ekarD ta gnikool em si emeM naM redipS ehT
em ro yerbuA ti si todK ti sI
tneve nonaC a stI
sraey fo snoillim rof no egar lliw taht erif a ni htrae eht flugne lliw
   msylcatac gnitluser eht dna edilloc lliw noom eht dna nuS ehT
C B ,A
A B C
BNJUYGBNJUYGBNMKIUYBNKUY
L A S D F G H J K L P O I U Y T R E W Q AA Z X C V B M M K H R 12 DFGHYTRED
   JUYTREDFGHJ REDCVBJUYTR SDCVBNJUYTRESDFHJ ETRD WED5TRED REFT54ER R54ERT54R
   !@#$%^&*()
```

Test Output To Screen: (*indented lines are continuation of previous line***)**
```
Hello Its Me
According to my calculations
Chuanli Says Give me an A
The Spider Man Meme is me looking at Drake
Is it Kdot is it Aubrey or me
Its a Canon event
The Sun and the moon will collide and the resulting cataclysm will engulf the
   earth in a fire that will rage on for millions of years
A, B C
C B A
YUKNBYUIKMNBGYUJNBGYUJNB
)(*&^%$#@! R45TRE45R RE45TFER DERT5DEW DRTE JHFDSERTYUJNBVCDS RTYUJBVCDER
   JHGFDERTYUJ DERTYHGFD 21 R H K M M B V C X Z AA Q W E R T Y U I O P L K J H G
   F D S A L
```

Problem #5
60 Points

5. Dominika

Program Name: Dominika.java **Input File: dominika.dat**

Test Input File:
```
11
-7 -11 -1 -5
2 -16 8 -5
9 19 17 -15
15 -4 -10 -8
-14 11 -10 -10
-17 -5 0 -15
-5 16 -7 -16
-5 -4 6 13
-5 1 -19 -2
-1 0 1 0
0 -1 0 1
```

Test Output To Screen:
```
TEST CASE: 1
(-9.20,-2.80)
(1.20,-13.20)
TEST CASE: 2
(-4.53,-5.30)
(14.53,-15.70)
TEST CASE: 3
(-16.44,-4.93)
(42.44,8.93)
TEST CASE: 4
(-0.96,15.65)
(5.96,-27.65)
TEST CASE: 5
(-30.19,-2.96)
(6.19,3.96)
TEST CASE: 6
(-17.16,-24.72)
(0.16,4.72)
TEST CASE: 7
(-33.71,1.73)
(21.71,-1.73)
TEST CASE: 8
(-14.22,14.03)
(15.22,-5.03)
TEST CASE: 9
(-14.60,11.62)
(-9.40,-12.62)
TEST CASE: 10
(0.00,-1.73)
(0.00,1.73)
TEST CASE: 11
(-1.73,0.00)
(1.73,0.00)
```

Problem #6
60 Points

6. Gerald

Program Name: Gerald.java **Input File: gerald.dat**

Test Input File:
```
4435
5652    4415    4368    4754    2858    6758    7743    1210    1047
4537    8714
9316    6834    9433    7647    3849    8348    5486    4673
7067    3668    3777
7956    7395    6116    6380    1783    9444    6683
4843    1050    7750    8991
9876    1837    1514    8536    5569    7563
2976    6998    6816    2297    9655    2339    7463    4805    1495
4191    5357    7013    8854    8199    1046    5319    6151    2851    3620
```

Test Output To Screen:
```
COUNT:59
SMALLEST:1046
LARGEST:9876
MEAN:5547.8
```

Problem #7
60 Points

7. Hiro

Program Name: Hiro.java   **Input File: hiro.dat**

Test Input File:
```
10
MCMIV
XIVII
IVI
MCMLXXXIV
LXXX
DCCLXXXVI
MCCXCVIII
MMXVII
CIX
XCVIII
```

Test Output To Screen:
```
1904
16
5
1984
80
786
1298
2017
109
98
```

Problem #8
60 Points

8. Joan

Program Name: Joan.java   **Input File: joan.dat**

Test Input File:
```
The Computer Science contest challenges students to study a broad range of areas
in computer science and has both an individual and a team component Competition
consists of a minute written exam for both components along with a two hour
programming contest for teams
At all levels of competition individual places are determined solely by written
exam scores All contestants compete for individual honors at all levels of
competition Individuals placing first second and third advance to the next level
For the team component the teams top three scores on the written exam are added
to its score in the programming session to produce an overall team score
First place teams advance to the next level of competition
The contest was established in the school year with several goals in mind
Among them were the goals of further strengthening the UniversityInterscholasticLeague
math and science program with a new technological component and to support Texas
public schools in meeting the growing demand for computer education
Stringingcollectionofwordywordywords justtomakeaveragelengthlongerandlonger
thanthestudentdatasample butitischallengingtosaytheleastbutnotthemost
Javaisthedesignatedprogramminglanguagefor
UniversityInterscholasticLeague
ComputerScienceandInformationTechnologyRulestheWorld
Finallygothere
```

Test Output To Screen:
```
172 words with an average length of 7 letters
```

Problem #9
60 Points

9. Manisha

Program Name: Manisha.java   **Input File: manisha.dat**

Test Input File:
```
12
WER +NO+ S / TR ANG E+ T/O+ LO V/E
YO U+KN OW+TH/E +R U/LE/S+A ND +S/O+DO +/I
/A+ /FUL+C /OMITE NS+W H/AT+IM+T HI/ NKG +OF
Y O/U+WO UL/DNT+G ET+T/HI S+F/ROM+ AN Y+OT/H ER+G/UY
I+ JUS T+WA/N+ TEL +YO/U +H/OW +IM+ F/E LI NG
G OTA+M A/KE+Y O/U +UN D/ERS T/A
NEV R+G O/N A+G/I VE+YO U+/UP
NEVR +G/ON A+L/ET+ YOU +D/O WN
NE VR+G O/NA+R U/N +A RO/U ND +A/ND +DE S/RT +Y/O U
N E/VR+G /ONA+ /MAK E+YO U+C/RY
N E/VR+ GO/NA +S /AY+ GO DBY/E
N E/VR+ GO/NA+T E/L+A +L/IE +A/ND +H/UR T+ YO/U
```

Test Output To Screen:
```
NO+/OT+ERW+/ELOV+AEGNRS/T
/I+DO+/OS+ADN+/EHT+OUY+KNOW+E/LR/SU
/A+FO+IM+/FLU+/AHTW+GHIK/NT+CEIMN/OST
ANY+EGT+G/UY+O/UY+FMO/R+/HIST+E/HORT+/DLNOTUW
I+IM+A/NW+ELT+H/OW+O/UY+JSTU+/EFGILN
O/UY+AE/KM+AGOT+/AD/ENRSTU
P/U+OUY+AG/NO+EG/IV+ENRV
/ELT+OUY+AGN/O+DN/OW+ENRV
AD/N+/NRU+/OUY+AG/NO+ENRV+DE/RST+ADNOR/U
C/RY+OUY+AEK/M+AGN/O+ENR/V
/ASY+AG/NO+ENR/V+BD/EGOY
A+AD/N+E/IL+E/LT+O/UY+AG/NO+ENR/V+HRT/U
```

Problem #10
60 Points

10. Patryk

Program Name: Patryk.java   **Input File: patryk.dat**

Test Input File:
```
50
53334 82529 47619 40058 90010 23633 60058 49703 62205 56691
73206 11966 77917 73727 20058 58657 68587 45595 73853 67357
72704 68846 63854 86891 14006 22229 57124 96433 11027 84600
56052 29903 28595 31476 18155 26546 25819 23378 52995 35956
76081 15970 43909 95754 61556 97443 76534 41402 85527 14323
```

Test Output To Screen:

84600	*Continued from previous column…*
90010	14006
15970	73206
76081	26546
56691	68846
86891	61556
41402	35956
56052	11966
49703	31476
29903	77917
14323	11027
96433	85527
23633	73727
97443	67357
73853	58657
72704	68587
57124	20058
53334	40058
76534	60058
95754	23378
63854	43909
62205	47619
18155	25819
45595	22229
28595	82529
52995	

Continues next column…

Problem #11
60 Points

11. Rodrigo

Program Name: Rodrigo.java   **Input File: rodrigo.dat**

Test Input File:
```
12 20 2500
Prague 4.3 321.43 9 2.7
Milan 3.7 456.70 9 4.2
Casablanca 2.6 134.52 5 3.8
Paris 1.2 1000.03 10 2.5
Beijing 3.5 300.54 7 3.9
Bogota 2.4 140.23 3 4.8
Mexico_City 1.9 249.85 6 4.9
Moscow 1.2 333.33 8 1.2
Cape_Town 3.4 305.68 7 3.0
Sydney 3.6 278.23 8 1.3
Tokyo 4.3 412.39 10 3.9
Dubai 3.3 189.43 4 2.4
Prague Milan 400
Milan Casablanca 700
Prague Paris 550
Casablanca Paris 700
Dallas Paris 1400
Dallas Casablanca 1300
Milan Beijing 2100
Casablanca Beijing 2000
Bogota Dallas 650
Mexico_City Dallas 350
Mexico_City Bogota 350
Bogota Casablanca 1100
Prague Cape_Town 1200
Cape_Town Casablanca 900
Tokyo Sydney 500
Tokyo Dallas 2500
Sydney Beijing 700
Casablanca Dubai 750
Dubai Sydney 1400
Cape_Town Dubai 1800
Moscow Prague 900
Beijing Moscow 2300
Moscow Dallas 4000
Mexico_City Milan 1400
Paris Bogota 1800
```

Test Output To Screen:
```
1: Tokyo
2: Paris
3: Milan
4: Prague
5: Cape_Town
6: Mexico_City
7: Casablanca
8: Dubai
9: Bogota
```

Problem #12
60 Points

12. Svetlana

<table>
<tr><td>Program Name: Svetlana.java</td><td>Input File: svetlana.dat</td></tr>
</table>

Test Input File:

```
5
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ _ _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ _ b _ _ _ _
_ b _ _ _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ _ _ _ _ _ _ _
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ _ b _ _ b _
_ b _ b _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ B _ r
r _ r _ _ _ r _
_ r _ _ R _ _ r
_ _ R _ _ _ _ _
_ _ _ b _ _ _ _
_ _ _ b _ _ b _
_ b _ b _ _ _ _
b _ b _ b _ R _
----------------
_ r _ r _ r _ r
r _ r _ r _ r _
_ r _ r _ r _ r
_ _ _ _ _ _ _ _
b _ b _ b _ b _
_ b _ b _ b _ b
b _ b _ b _ b _
----------------
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
```

Test Output To Screen:

```
Test case: 1
Red valid single moves: 10
Black valid single moves: 7
Test case: 2
Red valid single moves: 12
Black valid single moves: 10
Test case: 3
Red valid single moves: 14
Black valid single moves: 8
Test case: 4
Red valid single moves: 7
Black valid single moves: 7
Test case: 5
Red valid single moves: 0
Black valid single moves: 0
```